ONLY HERPES LAST FOREVER

by Burks A. Smith of Datasmith
PO Box 8036, Shawnee Mission KS 66208

(Ed. Note: This is a continuation of an article which started in last month's MUG newsletter.)

FROM MDOS AND CP/M TO MS-DOS

When I started working with microcomputers in 1978, I chose a Micropolis subsystem because of the excellent BASIC and a disk storage capacity that was about double other 5 1/4" disks and quite a bit more than an 8" disk. I used MDOS to write many useful programs in my manufacturing business. When I decided to become a full-time software designer, I began to assemble a second computer from the board level to save money. I bought the last "discontinued" IMSAI 8080 chassis from Computerland and built the processor and memory from Compupro kits. Naturally, it also had a Micropolis subsystem so it would be compatible with the first computer, and I chose a Vector Graphic Mindless Terminal and Bit-streamer I/O card so that I could use Vector Graphic software without modification. For the finishing touch, I even bought an official Vector Graphic monitor ROM, which allowed me to run screen oriented programs like the Scope editor and Raid de-bugger. I sold many programs for Vector computers, both in MDOS and CP/M. In 1982 I got an IBM PC. To date, I have already upgraded the first PC to a hard disk and bought a second PC-XT with a hard disk. 99% of my business is now for IBM and IBM compatible computers, but I use my Micropolis based machine every day for word processing, payroll, and bookkeeping. The oldest Micropolis based computer has a bad terminal and some flaky memory, but the disk drive still works fine and I am saving it as a backup.

Micropolis MDOS was written by a company in the disk drive business so that their customers would have something to do with the disk drives they were trying to sell. In those days, there wern't many micros in use, and CP/M had not been estabilised as the dominant operating system. Radio Shack's TRS-DOS and North Star DOS were very popular, not to mention APPLEDOS.

When Vector stopped shipping MDOS as standard on their machines, I learned how to use CP/M and Microsoft Basic and eventually transferred my commercial software to CP/M. When the PC came along, I again moved the software, and now have versions to run under three different operating systems. I use a serial data cable between the IBM PC and the Micropolis for routine transfers of files and have very little trouble keeping both machines busy. I haven't written a Micropolis Basic or MDOS program for several years, but I never did like CP/M that much. I think MS-DOS is great, but still have complaints about Microsoft BASIC.

If you get a new MS-DOS computer, you won't want to abandon your present computer. It can ease the transition, and with a little communication between the two computers, you can make use of much of your programs and data on the new machine. If you use MDOS, the format of your data will probably have to be changed before it can be used.

Software is a different problem. Since MS-DOS machines use a different microprocessor, any machine-language programs you have will not run on the new machine. If you write programs in assembly language, significant changes will have to be made in your source code to get programs to run on the new machine. BASIC is a different matter. If you are now using Microsoft Basic and CP/M, the changes in your program will be minor, if any, since PC-BASIC is a superset of the original Microsoft BASIC. If you use Micropolis basic, the translation will have to be more extensive, but is not too difficult.

FILE FORMAT DIFFERENCES

Micropolis disk drives, as you know, use 16 "hard sectored" disks and write approximately 256 bytes in each record. The number of tracks and sides varies with the type of system you have. MDOS is oriented toward the 256 byte record and is incapable of using different record sizes. If you want to maintain records of other than 256 bytes, you will have to write software that simulates it. The IBM PC uses 40 track "soft sectored" diskettes with either 8 or 9 sectors per track. Each sector is 512 bytes long, but both CP/M and MS-DOS handle records of a programmer-specified size, so the actual sector size on the disk is unimportant to the user. IBM double sided disks have a capacity that is approximately the same as a single-sided Micropolis Mod II disk, or about half the data density. The new IBM PC-AT, which was just announced, is reported to have new high density drives that surpass Micropolis in density, finally.

Data is stored by Micropolis BASIC in 256 byte records, which may hold a number of different variables. These variables are always read and written in the same order but may vary in length. Numbers stored consecutively on the disk are separated by at least one blank space, and strings are stored between "string delimiters", a special character defined by the STRING statement as being used for this purpose.

The file formats for GW-Basic are identical to the old version 5 conventions. There are two formats.

"Sequential" format has no record structure. The data is written in ASCII to disk as it would appear in a DATA statement, with commas or carriage returns separating individual data values. Sequential files may have new data written to the end of them, but must be read from the beginning to find a particular value. This is good file structure for communications and text purposes and is fast and easy to use.

"Random" format has fixed length records like Micropolis Basic, but the record length may be from 1 byte to 32K bytes. Data within a record may be organized in either a fixed field width or sequential format. The fixed field width format requires packing and unpacking records, which takes quite a bit of additional code. Numbers are usually stored in the internal binary format for very efficient use of storage. Sequential format within a fixed length record closely matches the Micropolis Basic method, except that the string delimiter is a comma or carriage return. Strings may be enclosed in quotation marks so that they may contain imbedded commas, but strings may not contain quotation marks.

SENDING FILES TO MS-DOS

Transferring data from MDOS computer to an MS-DOS computer is actually easier than transferring from MDOS to CP/M on the same machine. With two computers, each running its own operating system, there is no disk format translation necessary. Since the Micropolis disk format is hopelessly incompatible with the soft secored disks commonly used by the IBM PC and its clones, the best way to send data to the new machines is through a communications line. All you need is a serial port on each machine and an RS-232 cable for hooking them up. The transmission rate can be as high as 9600 baud (roughly 960 characters per second) with the right software.

If you already have some means of communication, such as a modem program, you can use it to transfer your files. If you don't have any communications software, you have several options. First, if you have the hardware and software to drive a serial printer like a Diablo or NEC, you can just "print" your files to the new computer. A simple Basic program on the MS-DOS machine can emulate a printer save your printed output to disk. Almost all Vector Graphic computers have this capability either through the CP/M "CONFIG" program or the MDOS "DIAB" overlay. Second, if you have a serial port but no printer driver for it, you could write a Basic program to read a file and send it to your serial port. A one-way transfer with no error checking needs only very simple logic. Third, you could buy some software. There are quite a few communications packages available for CP/M and MS-DOS, but not many for Micropolis MDOS. If you need an extremely reliable data transfer at high speed, the public-domain program known as "XMODEM" by Ward Christiansen can be obtained through most CP/M user groups at a small fee. The program is written in assembler and usually needs to be configured for a specific machine, so don't take this route if you're not a programmer. The MUG probably has something in the library that will work under MDOS.

Programming an IBM-PC or a PC-Compatible to simulate a serial printer is simplified by the fact that communications on these machines are interrupt driven and automatically buffered. This means that if data is detected coming in on a PC's serial line, the processor is interrupted from whatever it was doing for long enough to get the byte and store it away for later use. This holds true for any running program except timing sensitve operations like physical disk I/O and allows you to use Basic to write communications programs without worrying too much about the slow speed of Basic. In the examples below, sending output to a serial printer port is assumed.

Unless it is written in a high level language like Basic, a program can not be moved from MDOS to MS-DOS. The 8086 family of microprocessors does not have the same instruction set as the 8080 family, so 8080 machine language programs won't run on an 8086. If you write programs in assembly language, your source code will have to be substantially rewritten for the 8086, but the chips are close enough in architecture that no radical logical changes will be necessary.

Microsoft Basic is sufficiently close to Micropolis Basic that programs that work on one can be modified to run on the other. The "tokenized" disk formats of the two languages are totally incompatible so Micropolis Basic program files must be converted to ASCII text before they can be used by Microsoft. This is most easily done if you are using a printer driver to transmit data. Simply ASSIGN(2,3) and LIST the program with the other computer connected to the printer cable. Don't use LISTP unless you want the white space between pages to be sent too.

Data files containing plain text can be sent through whatever listing facility you have with the program that created the files. For example, a text file created by "LINEEDIT" can be dumped to the serial printer port with the PRINTP command. Likewise, files created by a word processer can use the word processor's printer list facility.

Data files created by a Basic program can be sent by writing a Basic program to read the files, reformat the records to suit Microsoft Basic, and send them to the serial printer port as if a report were being printed. The way you intend to use the data on the new computer dictates which kind of reformatting is necessary. The simplest format is the sequential file. In the following example, a Micropolis file containing variables A,B, and C is converted to a sequential file file that can be read by GW-Basic:

```
10 OPEN 1 "DATAFILE" END 60
20 OPEN 2 "$P"
30   GET 1 A,B,C
40   PUT 2 A;",";B;",";C
50 GOTO 30
60 PUT 2  CHAR$(26) ! eof mark
60 CLOSE 1: CLOSE 2
70 END
```

Note that the only reformatting that is necessary in this case is the addition of a comma delimiter between the variables. The carriage return and linefeed that separate individual records are valid delimiters in GW-Basic so no explicit comma is required after the last variable in a record.

As written above, the file created as MS-DOS stores the data is a "sequential" file that can not have random access by record. To create a file that you may access randomly, it must have a fixed record length and it is highly desirable to pack or "SET" numeric variables in binary format, fixed-length fields. Double precision values (about 15 digits accuracy) can be stored in 8 bytes, single precision (about 6 digits), can be stored in 4 bytes, and integer (+-32765) in 2 bytes.

The following program in GW-BASIC will read the file created above and convert it to a random access file containing 3 single precision values per record, or 12 bytes per record.

```
10 OPEN "DATAFILE" FOR INPUT AS #1
20 OPEN "DATAFILE.RND" AS #2 LEN=12
30 FIELD #2,4 AS A$,4 AS B$,4 AS C$
40 WHILE NOT EOF(1)
50   INPUT #1, A,B,C
60   LSET A$=MKS$(A): LSET B$=MKS$(B)
70   LSET C$=MKS$(C)
80   PUT #2
90 WEND
100 CLOSE 1
110 END
```

In the above example, line 10 could also be written as OPEN "I",#1,"DATAFILE", and line 20 could be written as OPEN "R",#2,"DATAFILE.RND", which is the Microsoft syntax from earlier versions. The FIELD statement allocates space for the packed data, and LSET and MKS$() do the packing. The PUT statement writes the data to disk in the next available record. The Microsoft program is longer than its Micropolis equivalent but is not restricted to a record size. Record size is specified in the OPEN statement.

MICROPOLIS TO MICROSOFT

Converting programs from Microsoft Basic to GW-Basic is a fairly simple process, but the languages have major differences in the way they handle files. If random access files are to be used, they should be carefully laid out before starting the translation. Write subroutines to read, write and FIELD so that Micropolis GET and PUT statements can be replaced by GOSUB's. GW-Basic is very fast at logic, math and program branches, but is slow in some kinds of I/O.
GW-Basic allows long variable names, so you are not restricted to the single letter or letter followed by a number in Micropolis. Multiple character labels add greatly to a program's readability, but keep in mind that scanning for excessively long variable names is slower than scanning for short ones.

The best tool for early translation is a good text editor or word processor. Start with the Micropolis Basic program in ASCII format and proceed as follows:

1. Find all the exclamation points used as REMs and replace them with an apostrophe, which is the REM equivalent in Microsoft Basic.

2. Change all CHAR$ to CHR$. Change REPEAT$ to STRING$ function, noting that STRING$ has a different syntax and will only repeat a single character. Change INDEX to INSTR.

3. Change all OPEN statements to correct syntax for the type of file being used. If the file is for random access, execute FIELD statements for the file. If the program has error handling, an ON ERROR GOTO statement must be included instead of the ERROR option of the OPEN statement. Note that ON ERROR GOTO intercepts all errors, even non-disk errors such as syntax errors. You may test the reserved variable ERR to determine which error has occrred, and variable ERL for the line number where it occurred. One little idiosyncracy is that you must CLOSE a file on a "file not found" error before you can use the file number again. The number of files that can be open at one time defaults at 3, but may be specified as any number when BASIC is loaded. Legal file numbers are from 1 to the largest number of files that may be open. When using the defaults, files may only be numbered 1, 2, and 3.

4. Find all the PUT statements and substitute a GOSUB to your disk output subroutine, perhaps passing a record number. If the PUT statement is for screen or printer output, change it to a PRINT # statement, which must be used for a sequential device like a printer, keyboard, or terminal.

5. Locate all the FMT functions. Microsoft has no FMT function, and does formatted output by use of the USING clause in a PRINT or PRINT # statement. You must change any PRINT statement containing a FMT function to a PRINT USING statement.

6. Change screen and cursor control code to that of the new system. Cursor control may be accomplished by PRINTing control codes, but GW-Basic has statements like LOCATE, COLOR, SCREEN, CLS, and BEEP that provide more sophisticated control of the terminal.

7. Save the edited text, get BASIC up, and load your program, and type RUN. Syntax error? You're on your own.

Numeric precision is GW-Basic's weak point. Micropolis variable precision with the SIZES statement is unique, and give Micropolis Basic more accuracy in significant figures than most mainframes. Micropolis also stores numbers in decimal (BCD) format, which is preferred for financial calculations.

Microsoft has four types of variables; integer, single precision, double precision, and string. Integers are simply signed 16-bit binary numbers in the range of plus or minus 32767. They excel as counters, flags, and array subscripts because they are very fast. Single precision and double precision variables store floating-point binary numbers with accuracy of about 7 and 15 significant figures respectively when converted to decimal. I have encountered instances where, because of differences in notation and conversion, GW-Basic finds it impossible to store a number I have entered exactly. A number entered as dollars and cents can sometimes be turned into a decimal with 13 numbers or so to the right of the decimal point. This small degree of error can cause bugs in programs. For example, if a double-precision number is subtracted from another double-precision number of apparently the same value, the answer may not be exactly zero, and would not test equal to zero in an IF statement.

String variables may be from 0 to 255 characters in length and have space dynamically allocated to them. Therefore, you do not have to dimension long strings as if they were arrays and can leave out the length dimension on a string array.

Some time spent reading the manual will alert you to many of the things you should look for when translating you program.

RECEIVING A FILE ON AN IBM PC

The following program emulates a serial printer's protocol so that an IBM PC (and most other GW-Basic machines) can save files to disk by appearing to your Micropolis-based computer as if it were a printer. This protocol matches that expected by Vector Graphic printer driver software in the MDOS "RES" overlay called "DIAB" and in the CP/M "CONFIG" program as "Qume/Diablo". Other computers will have the same protocol when dealing with these same printers.

In order to prevent the serial printer's buffer from overflowing, the printer drivers use a paced transmission method called ETX/ACK protocol. After outputting a block of data, usually a CR-terminated line, the computer sends an end-of-text (ASCII 3, called ETX) character to the printer and goes into a wait loop where further transmission is suspended. When the printer catches up to the point where it finds the ETX in its buffer, it responds by sending an acknowledge (ASCII 6, called ACK) character back to the computer signifying that all characters up to the ETX have been printed. Upon receiving the ACK, the computer resumes transmission for another line of data.

In the emulator program, Basic buffers all characters received by appending them to a string until it gets the ETX character. Knowing that the sending computer won't be sending anything else until it gets the ACK, the program takes time out to write the received line to disk and then responds with the ACK. This process continues until the emulator receives a control-Z (CHAR$ (26)), which is the sequential end-of-file mark for CP/M, MS-DOS, and all versions of Microsoft Basic. When the control-Z is received, the emulator program closes the file and asks for new file name. If none is entered, the program stops.

Usually, Basic isn't a good choice for a communications program because it is relatively slow and there is a chance that characters could be lost if the transmission rate (Baud Rate) is too high. However, since the IBM PC has an interrupt-driven communications port, it will buffer characters received until Basic has a chance to read them. However, a disk write can not be interrupted because it requires the processor's undivided attention to maintain the proper timing. This is why the ETX/ACK protocol is necessary to avoid possible loss of data. In our office, the program is used routinely at 9600 baud, but you should select the baud rate used by your serial port, usually 1200 for serial printers.

```
10 ' ### XCOM ###
20 'SERIAL PRINTER EMULATOR WITH ETX/ACK PROTOCOL
30 ' For IBM PC
40 ' Datasmith
50 ' Box 8036, Shawnee Mission KS 66208
60 '
70 ON ERROR GOTO 590
80 PRINT "SERIAL PRINTER EMULATOR"
90 ETX$=CHR$(3):    ACK$=CHR$(6)
100 CAN$=CHR$(24)
110 FALSE%=0: TRUE%=NOT FALSE%
120 INPUT "Enter baud rate";BAUD%
130 GOSUB 400 'INITIALIZE COMMUNICAITONS
140 '
150 ' <RECEIVE A FILE (DTE ONLY)
160 '
170 INPUT "ENTER NAME OF FILE TO RECEIVE";FILE$
180 IF FILE$="" THEN 640 'EXIT IF NO NAME ENTERED
190 OPEN "O",#1,FILE$
200 '< FILE TRANSFER LOOP
210 IF INKEY$=CAN$ THEN A$=CAN$: GOSUB 540: PRINT "CANCELLED": GOTO 370
220 GOSUB 490: IF B$="" THEN 210
230 IF B$=ETX$ THEN 280
240 IF B$=CHR$(26) THEN 350
250 REC$=REC$+B$
260 IF NOT RT% THEN IF B$<>CHR$(10) THEN PRINT B$;
270 GOTO 200
280 '<WRITE A REC
290 IF RT% THEN PRINT ".";
300 PRINT #1,REC$;
310 A$=ACK$
320 GOSUB 540
330 REC$=""
340 GOTO 200
350 PRINT: PRINT "END OF FILE ON ";FILE$
360 BEEP
370 CLOSE #1
380 GOTO 150 'REPEAT
```

```
390 '
400 ' < INITIALIZATION communications file
410 '
420 COMFILE$="com1:"+STR$(BAUD%)+",n,8"
430 OPEN COMFILE$ AS #2
440 KEY OFF
450 LOCATE ,1,1
460 IF BAUD%<1200 THEN RT%=FALSE% ELSE RT%=TRUE%
470 IF LOC(2)<>0 THEN B$=INPUT$(LOC(2),#2): B$="" 'FLUSH BUFFER
475 RETURN
480 '
490 '< INPUT CHAR (IF ANY) FROM SERIAL PORT AND RET IN B$
500 '
510 IF EOF(2) THEN B$="" ELSE B$=INPUT$(1,#2)
520 RETURN
530 '
540 '< OUTPUT A$ TO SERIAL PORT
550 '
560 PRINT #2,A$;
570 RETURN
580 '
590 ' ERROR HANDLER
600 '
610 IF ERR=57 THEN PRINT CHR$(233);: RESUME
620 ON ERROR GOTO 0
630 '
640 ' < STOP PROGRAM
650 END
```

```
##########################################################################
#                                                                        #
#                    BASIC/Z SPECIAL INTEREST GROUP                      #
#                                                                        #
##########################################################################
```

BASIC/Z CORNER #21

by Steven Guralnick
375 South Mayfair Avenue, Suite 205, Daly City, CA 94015

Those of you who have been following this column for a while know that I have produced, and am selling, a tickler program and a payables program.

I decided that I wanted to tie a serial number to the program, to track the origin of it. Normally, this can be done by implanting it into some data file. In this case, that wasn't practical because there is no data file that is never changed. TICK.DAT on the tickler program and PAY.DAT on the payables program are the only data files and they can (and usually are) erased when the user is finished playing with the demo data on them. Any serial number stashed in there would be lost unless I reserved the first record.

So, I spoke with System/z and asked them how I might plant a serial number either into the .COM file or into the .BZO file before BINDing. To my surprise, I learned that it is possible to open up the .COM file, after it has been created with the BIND command.

What you do is to place a target in the source code, as a literal string, such as CODE$="STEVEN G.". Then, using your serial implanter, you swap your serial number for the target. It has to be a byte for byte swap. In other words, if the target is ten bytes, the serial number has to be ten bytes.

Because the literal serial number can be looked at, intact, with any kind of dump program, it is desireable to scramble the number before implanting it and also to put some kind of checker elsewhere in the destination program, using another target.

What I will show you today is the basic implant routine. For obvious reasons, I will omit the scrambling routine and the checker routine. However, it gets pretty obvious.

```
10 REM THIS IS THE SERIALIZER ROUTINE FOR SERIALIZING COM FILES
20 ON ERROR GOTO @GLOBAL.ERROR
30 REM
40 UPCASE
50 STRING ""
60 DIM A$(128)
70 IF COMMAND$="TICKLER" OR COMMAND$="PAY" THEN FLNAME$=COMMAND$+".COM"
80 IF COMMAND$<>"TICKLER" THEN IF  COMMAND$<>"PAY" THEN  CLS:PRINT "Include  the name  of the  program to  be serialized
(""TICKLER"" or ""PAY"")":CLOSE:STOP
90 CLS:PRINT TAB(20);"$$$ SERIAL IMPLANT PROGRAM FOR ";FLNAME$;" $$$"
100  PRINTOUT$="":PRINT:PRINT"Do  you want  a printout  of remaining  serial numbers  (Y/N (<cr>  is no)  ";: PRINTOUT$=
EDIT$(1):PRINT "":GOSUB @ABORT
110 IF PRINTOUT$="Y" THEN CLS:GOTO @PRINTOUT
120  CLS:PRINT "What  drive will  hold the  disks to  be serialized?";:DRIVE$="":DRIVE$=EDIT$(1):PRINT  "":GOSUB @ABORT:
DRIVE$=DRIVE$+":"
130 REM
140 FILENAME$=DRIVE$+FLNAME$
150 LOWCASE:PRINT "Enter the serial number leadin";:LEADIN$=EDIT$(LEADIN$,4):PRINT "":UPCASE:GOSUB @ABORT
160 IF LEN(LEADIN$)<4 THEN LEADIN$=REPEAT$(" ",4)+LEADIN$:LEADIN$=RIGHT$(LEADIN$,4)
170 SERIAL.NUMBER.TARGET$="ROGER S.T.":REM THIS IS THE TARGET IN THE COM FILE THIS PROGRAM LOOKS FOR
180 REM
200 REM
230 PRINT"Mount the disk to be serialized in drive ";LEFT$(DRIVE$,1);" and press <cr>"
240 PRINT "or Press ESCape key to abort ";:CONTINUE$="":CONTINUE$=EDIT$(0):PRINT "":GOSUB @ABORT
250 RESET
260 OPEN 1 FILENAME$ UNFMT RECLEN 128 END 450 ERROR @CANNOT.FIND.THE.COM.FILE
270 COUNTER#=1
280 OPEN 2 "SERIAL.DAT" RECLEN 6 ERROR @CANNOT.FIND.THE.DATA.FILE
290 IF SIZE(2)=0 THEN CLS:PRINT "Serial numbers have been used up.":STOP
300 REPLACEMENT.SERIAL.NUMBER$=FNGET.THE.SERIAL.NUMBER$
310 REM
320 REM
330 REM
340 FOR COUNTER#=170 TO 1000:REM FOR A .COM FILE, START AT RECORD 170--FOR A .BZD  FILE, START  AT RECORD  1.  THE FIRST
170 OR SO RECORDS IN THE .COM FILE ARE THE RUN TIME LIBRARY.
350 GET  1 RECORD COUNTER# A$
360 PRINT TAB(12,39);COUNTER#
370 A&=INDEX(A$,SERIAL.NUMBER.TARGET$)
380 IF A&>0 THEN  A$=LEFT$(A$,A&-1)+SERIAL.NUMBER$+RIGHT$(A$,128-(A&+9)):PRINT TAB(20,10);"SERIAL  NUMBER ";REPLACEMENT.
SERIAL.NUMBER$;" IMPLANTED":PUT 1 RECORD COUNTER# A$:SERIAL.FLAG&=1
390 REM
400 REM
410 REM
420 IF SERIAL.FLAG&=1 THEN SERIAL.FLAG&=0: GOTO @QUIT:REM QUIT--NUMBER IS IMPLANTED
440 NEXT COUNTER#
450 IF SERIAL.FLAG&<>1 THEN PRINT"Cannot locate the serial number  target. Serial  number ";REPLACEMENT.SERIAL.NUMBER$;"
will not be used. Note this."
490 LABEL @QUIT:REM$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
500 CLOSE:PRINT TAB(23,10);"Finished.  Press <cr> to go  again or  ESCape to  exit";:CONTINUE$="":CONTINUE$=EDIT$(0 ):IF
ALTKEY<>1 THEN CLS:GOTO 130
510 PRINT TAB(23,70);"ALL DONE":CLOSE:STOP
520 REM
```

```
530 REM
710 DEF FNGET.THE.SERIAL.NUMBER$:REM****************
720 REM THIS ROUTINE GETS THE NEXT FILE NUMBER FROM THE DATA FILE  AND THEN  DECREMENTS THE  VARIABLE WHICH  BECOMES EOF
WHEN THE FILE IS CLOSED
730 NUMBER.OF.DATA.RECORDS=SIZE(2)
740 GET 2 RECORD NUMBER.OF.DATA.RECORDS SER.NO
750 DECR NUMBER.OF.DATA.RECORDS
760 EOF(2)=NUMBER.OF.DATA.RECORDS
770 FNEND$=LEADIN$+FMT(SER.NO,"999999")
780 LABEL @GLOBAL.ERROR:REM**************************
790 CONSOLE
800 PRINT "Line is ";ERRLINE
810 PRINT "Error is ";ERR$:GOTO 840
820 LABEL @CANNOT.FIND.THE.DATA.FILE:CLS:PRINT"I cannot locate the serial numbers file called ""SERIAL.DAT""":GOTO 840
830 LABEL @CANNOT.FIND.THE.COM.FILE:CLS:PRINT"I cannot locate the file called ";FLNAME$;" on drive ";LEFT$(DRIVE$,1)
840 CLOSE
850 STOP
860 LABEL @ABORT:REM*********************************
870 IF ALTKEY<>1 THEN RETURN
880 CLS:PRINT "Serial program aborted.  Shutting down.":CLOSE:STOP
890 RETURN
900 LABEL @PRINTOUT:REM******************************
910 REM THIS ROUTINE PRINTS OUT THE REMAINING SERIAL NUMBERS
920 HARDCOPY$="":PRINT "Hardcopy? (Y/N) (<cr> is no) ";:HARDCOPY$=EDIT$(1):PRINT "":GOSUB @ABORT
930 IF HARDCOPY$="Y" THEN CONTINUE$="":PRINT "Get the printer ready and press <cr>  ";:CONTINUE$=EDIT$(0):PRINT "":GOSUB
@ABORT
940 IF STATUS(2)=0 THEN OPEN 2 "SERIAL.DAT" RECLEN 6
950 IF HARDCOPY$="Y" THEN LPRINTER
960 PRINTOUT.COUNTER#=SIZE(2)
970 WHILE PRINTOUT.COUNTER#>=1
980 GET 2 RECORD PRINTOUT.COUNTER# A$
990 IF HARDCOPY$="Y" THEN PRINT FMT(VAL(A$),"____-999999")+" "+REPEAT$("_",50):PRINT  ELSE PRINT A$,
1000 GOSUB @PAGECHECK
1010 IF HARDCOPY$="Y" THEN GOSUB @PRINT.STOP
1020 IF PLINE>=50 THEN FORMFEED
1030 DECR PRINTOUT.COUNTER#
1040 WEND
1050 CONSOLE
1060 CLOSE
1070 IF HARDCOPY$="Y" THEN FORMFEED ELSE PRINT ""
1080 PRINT "Press <cr> to continue or ESCape to exit ";:CONTINUE$=EDIT$(0):PRINT "":GOSUB @ABORT
1090 GOTO 90
1100 LABEL @PAGECHECK:REM****************************
1110 REM THIS ROUTINE CLEARS THE SCREEN IF IT OVERLOADS--IT ALSO IS A QUICK STOP WHEN YOU PRESS THE ESCAPE KEY
1120 IF HARDCOPY$="Y" THEN RETURN:REM SKIP THIS IF THE PRINTER IS ON
1130 SCREEN.MAX&=PEEK(SETUP+63):REM GET THE LINES ALLOWABLE ON THE SCREEN FOR THIS TERMINAL
1140  IF  CLINE>=SCREEN.MAX&-3  THEN  BELL(1):PRINT  TAB(22,1);SPC$(78);TAB(22,15);"Press <cr>  to continue";:CONTINUE$=
EDIT$(0):IF ALTKEY=1 THEN CLOSE:CLS:CLRSUB:GOTO 90 ELSE PAGECHECK.FLAG&=1:CLS:RETURN
1150 RETURN
1160 LABEL @PRINT.STOP:REM***************************
1170 REM THIS ROUTINE STOPS OUTPUT TO THE PRINTER
1180 IF INSTAT=0 THEN RETURN
1190 IF INCHAR$<>CHR$(27) THEN RETURN
1200 CONSOLE
1210 BELL(2):CLS:CONTINUE$="":PRINT TAB(22,10);"Printer output is halted. Press <cr>  to continue  or ESCape  to exit.":
CONTINUE$=EDIT$(1):IF ALTKEY=1 THEN 1230
1220 FOR SCREEN.LINE&=22 TO 23:PRINT TAB(SCREEN.LINE&,1);SPC$(78):NEXT SCREEN.LINE&:LPRINTER:RETURN
1230 CLOSE:CLRSUB:GOTO 90
```

Herewith a few highlights. The program assumes you have a data file out there called SERIAL.DAT. How that is created

will follow at the end of this.

You turn on the program by typing SERPLANT, followed by the name of the program, in this case either TICKLER or PAY.

The program then picks up the name of the SERIAL. NUMBER.TARGET$, which I have set up as the same number of characters as the serial number, with LEADIN$. LEADIN$ is a four digit number you key in when you serialize the disk. So, you might end up with a number called 1234987654, ten bytes which will show up on the printout on the screen in the destination program (when it signs on) as 1234-987654. LEADIN$ is optional, the idea is to tell you where you sent the disk.

Then, you open up the .COM file at line 260, as a random record file, using the UNFMT option. This is what was a surprise. I didn't realize BASIC/z would allow me to read a hex file. System/z recommends using 128 byte record length. Shut off the delimiters(at line 50), DIMension the record as A$(128) (at line 60).

Then, start into a loop (at 340) and start reading one record at a time. Use the INDEX command at 370 to find the target. If you do, replace the target (which is at 170) with the latest serial number which is obtained from the function at 710. The serial number is turned into a string (at line 770) and the size of SERIAL.DAT is reduced by one, with the EOF command. For that reason, the serial numbers in SERIAL.DAT are put in there backwards, that is, for numbers from 1 to 100, 100 is first and 1 is last.

Once the serial number target has been replaced with the real number, turn it off (at line 420).

In the destination file, you have to look at the variable you established there and then you can see the number on the screen.

Here's the routine for creating the serial numbers file.

```
100 ON ERROR GOTO @GLOBAL.ERROR
200 REM THIS ROUTINE SETS UP THE SERIAL DATA FILE FROM WHICH SERPLANT WILL GET THE NUMBERS  FOR IMPLANTING  ON SOMEONE'S
DISK
300 CLS
310 RESET
400 GOSUB @CREATE.SERIAL.NUMBERS
600 GOTO @QUIT
700 LABEL @CREATE.SERIAL.NUMBERS:REM$$$$$$$$$$$$$$$$
800 REM THIS ROUTINE CREATES THE NUMBERS AND PUTS  THEM TO A DISK FILE CALLED SERIAL.DAT
900 PRINT "Serial numbers to be created start where ";:START.SERIAL.NUMBER$=EDIT$(6):PRINT "":IF ALTKEY=1 THEN @QUIT
1000 PRINT "Serial numbers to be created end where ";:END.SERIAL.NUMBER$=EDIT$(6):PRINT "":IF ALTKEY=1 THEN @QUIT
1100 CLRNONE
1200 CREATE 1 "SERIAL.DAT" RECLEN 6 ERROR 1300:GOTO
1400
1300 SCRATCH "SERIAL.DAT":GOTO 1200
1400 ON ERROR 1 CLEAR
1500 SERIAL.COUNTER=VAL(END.SERIAL.NUMBER$)
1600 WHILE SERIAL.COUNTER>=VAL(START.SERIAL.NUMBER$)
1700 PUT 1 SERIAL.COUNTER
1800 GOSUB @ABORT
1900 PRINT TAB(12,10);"WRITING SERIAL NUMBER";SERIAL.COUNTER
2000 DECR SERIAL.COUNTER
2100 WEND
2200 CLOSE
2300 RETURN
3700 LABEL @GLOBAL.ERROR:REM$$$$$$$$$$$$$$$$$$$$$$$$
3800 PRINT "Line is ";ERRLINE
3900 PRINT "Error is ";ERR$
4000 CLOSE
4100 STOP
4200 LABEL @ABORT:REM$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
4300 IF INSTAT=0 THEN RETURN
```

```
4400 IF INCHAR$<>CHR$(27) THEN RETURN
4500 CLS:PRINT "Press <cr> to continue or ESCape to quit ";:CONTINUE$=EDIT$(0):IF ALTKEY=1 THEN CLOSE:STOP
4600 CLS:RETURN
4700 LABEL @QUIT:REM##################################
4800 CLOSE:STOP
```

See you soon!

(End - Basic/z SIG)

##########

CLASSIFIED
==========

HELP: Some help in figuring out why my Micropolis CP/M (Computer Mart/Prodigy) won't boot on my IMSAI unless I've previously booted up through MDOS. Without first booting MDOS, CP/M starts to read, then flips back and forth, forever, between drives A: and B:. Bob Fortune, 240 SE 8th, Dundee OR 97115 - (503) 246-5705.

WANTED: A VG 3005 in used, but good condition. Also interested in the possibility of buying an 8" drive for the D: position. Guy Cusumano, AER-X-DUST, PO Box 93, Tennent NJ 07763 - (201) 431-1505.

HELP: Can someone help me find a repair shop for my IMSAI box & its cards? Gary Fenical, Instrument Specialties, Inc., Delaware Water Gap PA, 18327 - (717) 424-8510.

WANTED: Interested in buying a used (or new) Micropolis 1042 MOD I drive. Kirby Boswell, (215) 779-0522.

##########

I'm sure you're aware that it's getting to that Christmas shopping time. DAMAN may be able to help - especially with those "hard-to-get" items specifically for a Micropolis or Vector Graphic hard-sectored drive. DAMAN can get any software and convert it to 16 hard-sector format. If it isn't listed below, give us a call.

CP/M SOFTWARE
=============

CP/M Operating System (Lifeboat)....	$213.75	BASIC/Z - Compiling Language.....	$300.00
CATALOG - Disk Filing System.....	$ 62.51	POWER - System Utility.....	$137.52
CONDOR-1 - Data Base System.....	$240.73	CONDOR-3 - Relational Data Base.....	$525.35
dBASE II - Data Base System.....	$525.35	MINI-LEDGER - Home Accounting System	$126.35
WORD STAR - Word Processor.....	$388.36	SPELLBINDER - Word Processor Plus...	$304.57
BDS-C - Compiling Language.....	$126.35	ASSM PLUS TOOLS.....	$168.00
BOOKKEEPING.....	$239.40	PAYROLL.....	$319.20
SUPERCALC I.....	$161.10	SUPERCALC II.....	$223.90
INFO STAR - Data Base Manager.....	$391.02	WORD MASTER - Text Editor.....	$119.70
ANALIZA II - Psychiatrist Game.....	$ 43.89	HYPER TYPER - Typing Tutor.....	$ 42.56

MDOS SOFTWARE
=============

BEM - Basic Expansion Module.....	$ 51.87	BASIC/Z - Compiling Language.....	$ 85.00
SORT/A - Assm Lang Sort for Basic...	$ 59.85	XREF - Basic Cross-reference Gen'r..	$ 67.83
DSM-1 - MDOS Disassembler.....	$ 51.87	UTL-1 - MDOS Utility Set.....	$ 75.81
MDOS Extensions.....	$ 39.90	DATABASE TWO.....	$ 39.90
INVENTORY ONE.....	$ 39.90	MODIFILE/MODIMATH (for database two)	$ 39.90
REACT - Action Item List.....	$ 39.90	WAMSORT - Assmy Lan. Sort for Basic.	$ 39.90
BOOKKEEPING.....	$199.50	PAYROLL.....	$279.30

BARE 1115 DRIVE

'BARE' MICROPOLIS 1115 DRIVE

NO POWER SUPPLY, NO CABINET

Use as replacement in
Vector MZ, System B, etc.

Single Sided: (MOD II or V)
List: $360.00
MUG: $302.40

Double Sided: (MOD IV or VI)
List: $447.00
MUG: $375.20

DAMAN, 604 Springwood Circle
Huntsville AL 35803 (205)881-1697

ACCESSORIES

SCOTCH (3M) 16-HARD-SECTOR DISKS (10)	
Single sided	$30.59
Double sided	$47.03
FLIP-N-FILE 50 5" DISK HOLDER	$26.60

CP/M GAMES

ZORK I	$44.76
ZORK II	$44.76
ZORK III	$44.76
DEADLINE	$52.88
STAR CROSS	$44.76
SUSPENDED	$52.88
WITNESS	$52.88
PLANETFALL	$52.88
ENCHANTER	$52.88
SORCERER	$52.88
INFIDEL	$52.88

DAMAN, 604 Springwood Circle
Huntsville AL 35803 (205) 881-1697

SINGLE 1115 SUBSYSTEM

ONE MICROPOLIS 1115 DRIVE

IN CABINET, WITH POWER SUPPLY

Use as replacement or
Add-on in all systems.

Single Sided: (MOD II or V)
List: $493.33
MUG: $414.40

Double Sided: (MOD IV or VI)
List: $580.00
MUG: $487.20

(Data cables extra - made to order)

DAMAN, 604 Springwood Circle
Huntsville AL 35803 (205)881-1697

MICROPOLIS PARTS & DOCUMENTATION

1000 SERIES DRIVE MAINTENANCE MANUAL	$47.50
1100 SERIES DRIVE MAINTENANCE MANUAL	$23.75
PDS (MDOS) VERSION 4.0 (single sided)	
Disks and manual	$71.25
Manual only	$47.50
Disks only	$23.75
PDS (MDOS) VERSION 4.1 (double sided)	
Disks and manual	$95.00
Manual only (same as 4.0)	$47.50
Disks only (with tech note)	$47.50
100 TPI ALIGNMENT DISK	
Single sided	$47.50
Double sided	$95.00
DIAGNOSTIC DISK (requires MDOS)	$47.50
HEAD PADS	$ .50
SAUNDERS MAGNALUBE	$ 4.28
DRIVE MOTOR/TACH ASSEMBLY	$36.10
DRIVE BELT a('-	$ 5.75
PRINTED CIRCUIT BOARD A (single)	$166.25
PRINTED CIRCUIT BOARD A (double)	$256.50
1071 DISK CONTROLLER	$350.00

Published Monthly by DAMAN, 604 Springwood Circle, Huntsville AL 35803
Subscription rates: North America; $18/year Elsewhere; $25/year, Airmailed

For Assistance with general information, MUG/Vector Graphic, DAMAN hardware & software, Micropolis Basic, Basic/z, Micropolis drives & Micropolis parts-

Call Lynn or Buzz at (205)881-1697 during the Central Times of 9 AM to 9 PM.

PRINTERS

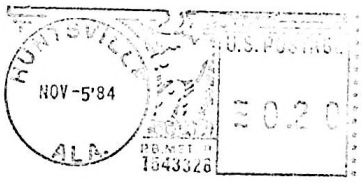
Guaranteed satisfaction. Return within 30 days for full refund.

Printer	List Price	DAMAN Price-1	DAMAN Price-2
Stx-80	$199	$177	$182
Gemini 10X	$399	$328	$338
Gemini 15X	$549	$448	$461
Delta 10	$549	$443	$456
Delta 15	$799	$645	$665
Radix 10	$849	$723	$745
Radix 15	$995	$845	$870
Powertype	$499	$426	$439

Accessories	List Price	DAMAN Price-1	DAMAN Price-2
Gemini Serial Interface	$ 59	$ 56	$ 58
Serial Interface with 4K Buffer	$119	$111	$114
Serial Printer Interface Cable		$ 25	$ 26

Price-1 = Pre-paid. Price-2 = Charge.

MICROPOLIS/VECTOR GRAPHIC
USERS GROUP
A Division of DAMAN
604 Springwood Circle
Huntsville AL 35803
(205) 881-1697



FIRST CLASS MAIL

FIRST CLASS MAIL